Tcl8.6.14/Tk8.6.14 Documentation > Tcl Commands, version 8.6.14 > fileevent

Tcl/Tk Applications | Tcl Commands | Tk Commands | [incr Tcl] Package Commands | SQLite3 Package Commands | TDBC Package Commands | tdbc::mysql Package Commands | tdbc::odbc Package Commands | tdbc::postgres Package Commands | tdbc::sqlite3 Package Commands | Thread Package Commands | Tcl C API | Tk C API | [incr Tcl] Package C API | TDBC Package C API

NAME

fileevent — Execute a script when a channel becomes readable or writable

SYNOPSIS

fileevent *channelId* **readable** ?*script*?
fileevent *channelId* **writable** ?*script*?

DESCRIPTION

This command is used to create *file event handlers*. A file event handler is a binding between a channel and a script, such that the script is evaluated whenever the channel becomes readable or writable. File event handlers are most commonly used to allow data to be received from another process on an event-driven basis, so that the receiver can continue to interact with the user while waiting for the data to arrive. If an application invokes **gets** or **read** on a blocking channel when there is no input data available, the process will block; until the input data arrives, it will not be able to service other events, so it will appear to the user to “freeze up”. With **fileevent**, the process can tell when data is present and only invoke **gets** or **read** when they will not block.

The *channelId* argument to **fileevent** refers to an open channel such as a Tcl standard channel (**stdin**, **stdout**, or **stderr**), the return value from an invocation of **open** or **socket**, or the result of a channel creation command provided by a Tcl extension.

If the *script* argument is specified, then **fileevent** creates a new event handler: *script* will be evaluated whenever the channel becomes readable or writable (depending on the second argument to **fileevent**). In this case **fileevent** returns an empty string. The **readable** and **writable** event handlers for a file are independent, and may be created and deleted separately. However, there may be at most one **readable** and one **writable** handler for a file at a given time in a given interpreter. If **fileevent** is called when the specified handler already exists in the invoking interpreter, the new script replaces the old one.

If the *script* argument is not specified, **fileevent** returns the current script for *channelId*, or an empty string if there is none. If the *script* argument is specified as an empty string then the event handler is deleted, so that no script will be invoked. A file event handler is also deleted automatically whenever its channel is closed or its interpreter is deleted.

A channel is considered to be readable if there is unread data available on the underlying device. A channel is also considered to be readable if there is unread data in an input buffer, except in the special case where the most recent attempt to read from the channel was a **gets** call that could not find a complete line in the input buffer. This feature allows a file to be read a line at a time in nonblocking mode using events. A channel is also considered to be readable if an end of file or error condition is present on the underlying file or device. It is important for *script* to check for these conditions and handle them appropriately; for example, if there is no special check for end of file, an infinite loop may occur where *script* reads no data, returns, and is immediately invoked again.

A channel is considered to be writable if at least one byte of data can be written to the underlying file or device without blocking, or if an error condition is present on the underlying file or device.

Event-driven I/O works best for channels that have been placed into nonblocking mode with the **fconfigure** command. In blocking mode, a **puts** command may block if you give it more data than the underlying file or device can accept, and a **gets** or **read** command will block if you attempt to read more data than is ready; a readable underlying file or device may not even guarantee that a blocking [read 1] will succeed (counter-examples being multi-byte encodings, compression or encryption transforms). In all such cases, no events will be processed while the commands block.

In nonblocking mode **puts**, **read**, and **gets** never block. See the documentation for the individual commands for information on how they handle blocking and nonblocking channels.

Testing for the end of file condition should be done after any attempts read the channel data. The eof flag is set once an attempt to read the end of data has occurred and testing before this read will require an additional event to be fired.

The script for a file event is executed at global level (outside the context of any Tcl procedure) in the interpreter in which the

fileevent command was invoked. If an error occurs while executing the script then the command registered with **interp bgerror** is used to report the error. In addition, the file event handler is deleted if it ever returns an error; this is done in order to prevent infinite loops due to buggy handlers.

EXAMPLE

In this setup **GetData** will be called with the channel as an argument whenever $chan becomes readable. The **read** call will read whatever binary data is currently available without blocking. Here the channel has the fileevent removed when an end of file occurs to avoid being continually called (see above). Alternatively the channel may be closed on this condition.

```
proc GetData {chan} {
    set data [read $chan]
    puts "[string length $data] $data"
    if {[eof $chan]} {
        fileevent $chan readable {}
    }
}

fconfigure $chan -blocking 0 -encoding binary
fileevent $chan readable [list GetData $chan]
```

The next example demonstrates use of **gets** to read line-oriented data.

```
proc GetData {chan} {
    if {[gets $chan line] >= 0} {
        puts $line
    }
    if {[eof $chan]} {
        close $chan
    }
}

fconfigure $chan -blocking 0 -buffering line -translation crlf
fileevent $chan readable [list GetData $chan]
```

CREDITS

fileevent is based on the **addinput** command created by Mark Diekhans.

SEE ALSO

fconfigure, **gets**, **interp**, **puts**, **read**, **Tcl_StandardChannels**

KEYWORDS

asynchronous I/O, blocking, channel, event handler, nonblocking, readable, script, writable.